REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-05-

0406

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 2, 2005	3. REPORT TYPE AND DATES COVERED Final Technial Report, 09/01/01 - 01/30/05

4. TITLE AND SUBTITLE Simulating the Interactions of Genes, Proteins, and Metabolites in Cell-Like Entities	5. FUNDING NUMBERS F49620-01-1-0505
6. AUTHOR(S) Brent D. Foy	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Wright State University 3640 Colonel Glenn Hwy Dayton, OH 45435-0001	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) DoD, Air Force Office of Scientific Research (AFOSR) NL	10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approve for Public Release: Distribution Unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The goal of this project was to create software to model Cell-Like Entities, and to perform initial simulations. A software package was written in Matlab that performs stochastic simulations using a Gillespie-based algorithm of biomolecular networks. Biochemical processes that can be modeled include binding, unbinding, transcription, translation, Michaelis-Menten, and n-th order processes. Models and parameters are defined using several text files. The software package was designed to enable simple porting to high-performance computing platforms. A model to simulate a prototype CLE that responds to an external chemical with production of a particular protein was developed. This detection CLE was further analyzed to quantify the level of stochastic fluctuations over multiple runs, and to determine a combination of input parameters that optimized the detection task.

14. SUBJECT TERMS			15. NUMBER OF PAGES 34
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

EXECUTIVE SUMMARY

- **Objective 1:** Create the software tools needed to simulate gene, protein, and metabolite interactions in a CLE.
  - A software program in Matlab was developed and transitioned to Air Force collaborators at Wright Patterson Air Force Base.
  - The program stochastically models biochemical processes including gene transcription, mRNA translation, and enzymatic, binding, and transport reactions.

- **Objective 2:** Using minimal cell ideas from the literature as a starting point, identify a set of genes and gene products that make a theoretically viable cell and conduct simulations which explore its viability under a range of external conditions.
  - Several models of a Cell-Like Entity (CLE) were developed. One demonstrates macroscopic effects of molecular level stochastic fluctuations, one explored two approaches to simulating a common enzymatic reaction, and one was designed to detect an external chemical.
  - These models were explored through simulations, but are not considered complex enough to test CLE viability.

- **Objective 3:** Add a specific designed function to the minimal cell and, through simulations, test whether this change altered the cell's viability and identify the conditions under which the task is successfully performed.
  - The level of stochastic fluctuations in the detection CLE has been measured under baseline conditions.
  - A determination of parameter values that enhance the performance of the task was performed.

- **Publications**
  - Karpinets TV, Foy BD. Tumorigenesis: the adaptation of mammalian cells to sustained stress environment by epigenetic alterations and succeeding matched mutations, Carcinogenesis, 26(8): 1323-1334, 2005.

  - Karpinets TV, Foy BD. Model of the developing tumorigenic phenotype in mammalian cells and the role of sustained stress, Journal of Theoretical Biology, 227: 253-264, 2004.

  - Karpinets TV, Foy BD, Frazier JM. Tailored Gene Array Databases: Applications in Mechanistic Toxicology. Bioinformatics, 20: 507-517, 2004.

  - Foy BD, Frazier JM. Incorporation of Protein Binding Kinetics and Carrier-Mediated Membrane Transport into a Model of Chemical Kinetics in the Isolated Perfused Rat Liver. Toxicology Mechanisms and Methods, 13: 53-75, 2003.

- **Personnel Involved:**
  - Dr. John Frazier, Senior Scientist, WPAFB.
  - Dr. Marvin Thrash, Scientist, WPAFB.
  - Dr. Tatiana Karpinets, Postdoctoral Scientist, Wright State University.
  - Christopher Geib, WPAFB.
  - Dinu Stoicovici, Masters Student, Physics Dept, Wright State University.
  - Angela Rae Blissett, NSF-REU undergraduate student, Wright State Univ.

- **Thesis**
  - Design and Optimization of the Chemical Sensing Capability of a Theoretical Biological Cell Using a Stochastic Simulation, Dino Stoicovici, Dept. of Physics, Wright State University, Masters Thesis.

TECHNICAL SUMMARY

CLE Simulation Software

The simulation software (called Biomolecular Network Simulator or BNS) is written in Matlab version 6.5 and uses the Gillespie stochastic algorithm (Gillespie DT, J Phys Chem, **81**: 2340-2361, 1977) to simulate a system of biochemical reactions in a volume. The BNS engine consists of 46 Matlab files contained in the main code directory. Additional files are used to define the model. We have used the software to create and run simulations of theoretical and experimental CLEs in cooperation with partners at WPAFB. Below I describe the key attributes of the BNS version 1.2 software. Appendix 1 has a roadmap of the files and functions called.

General Input Considerations

A model is defined in a series of text files that specify the names and initial amounts of each compound, the reactions, the rate constants for each reaction, and general constants such as the total duration of the simulation. The folder structure used to define a model will be helpful:

```
- BNS root directory
        46 engine files
        - model_0001 directory
                reactions_0001.m
                reactions_0002.m, etc
                - parameter_set_0001 directory
                        general_constants.m
                        storing_and_plotting.m
                        - initial_compound_numbers directory
                                cmpd_defs_0001.m
                                cmpd_defs_0002.m, etc.
                        - reaction_constants directory
                                reaction_constants_0001.m
                                reaction_constants_0002.m, etc.
                        - output directory
                                default_out.mat
                                output_0001.mat, etc.
                - parameter_set_0002 directory, etc.
        - model_0002 directory, etc
```

Any name with a '_0001' or '_0002' in it can be any text name desired – i.e. the '_0001' is not required. The other names are fixed.

This folder system separates the model reaction structure from the numerical constants. The model reaction structure is the definition of the connections between compounds, contained in the reactions_0001.m, reactions_0002.m, etc files. This part of a model could often be represented as a pictorial schematic of reactions.

The numerical constants are contained in the files and subdirectories under the *parameter_set_0001* directory. These constants include the initial compound numbers, reaction rate constants, general constants such as the simulation time and volume, and the list of which compounds to store and plot vs. time.

This folder structure was chosen so that one can archive the models and simulation results of multiple modeling sessions by multiple investigators. This is achieved by keeping the numerical constants and simulation output in a single directory that can be saved as a new parameter directory whenever changes to the constants are made. The same idea holds for multiple model directories as one changes the connections in the network being simulated.

After setting up the reaction structure and constants, the model is run by typing 'bns' on the Matlab command line while in the BNS root directory. The program then asks 4 questions: (1) name of model directory; (2) name of parameter directory; (3) name of output file; and (4) choice of random number generator seed. One is able to rerun a simulation with the exact same sequence of random numbers as a previous simulation if desired.

The general_constants.m file contains values for constants that affect the overall execution of the simulation. The following parameters are defined in this file through simple assignment statements (with some example values filled in). Comments are in italics.

time_info.time_start = 0;
time_info.time_end = 1000;

The beginning (usually zero) and end times for the simulation in seconds.

time_info.num_runs = 1;

The number of times to run the entire simulation.

time_info.init_timestep = 10;
time_info.timestep_max = 10;

The initial and maximum time step in seconds. The time chunks determined by these parameters are used for storing, plotting, and parallelization of execution.

scaling_factor = 1;

> *Scale the problem up in terms of volume and number of molecules in the simulation. Best to use positive integers in order to maintain integer numbers of molecules.*

volume = scaling_factor .* 8e-16;

> *The volume being simulated in m^3, multiplied by the scaling factor. Combined with the number of molecules, this determines the concentrations of the compounds.*

stochastic = 1;

> *If stochastic = 1, then use a stochastic algorithm. If stochastic = 0, then use a continuous, deterministic solver. The continuous solver tends to have problems with initial compound numbers of zero.*

stoch_algor = 1;

> *Choice of stochastic algorithm.*
>
> *1 = mine, which is a hybrid between the Gillespie First Reaction Method (Gillespie DT, J Comput Phys, **22**: 403, 1976) and the modifications proposed by Gibson and Bruck (Gibson MA and Bruck J, J Phys Chem A, **104**: 1876-1889, 2000).*
>
> *2 = Gillespie Direct Method.*

The *initial_compound_numbers* directory contains files such as cmpd_defs_001.m that specify the names of the compounds and the initial number of molecules of those compounds. Each file contains assignment statements of the following form:

```
ATP = 700000;
ADP = 4000:
```

There can be any number of files in the directory, and any number of compounds may be defined in each file. The separate files are merely to help organize the data. The names of the compounds must match the names of compounds used in the reactions_001.m files, and the program checks to ensure that this is true.

The details of specifying the connections in the biological network, specifying the output options, and specifying the reaction constants will be presented in the sections below.

Model Results – Output

The program can create plots during the simulation run and save the simulation results to disk. The storing_and_plotting.m file contains the following choices (with some example choices filled in).

```
plot_info.output_directory = [];
```

Directory to store the output from a simulation run. An empty matrix here will use a directory called 'output' under the parameters directory. Otherwise input the entire path to the (already present) output directory.

```
output_filename = 'default_out';
```

This next setting chooses a default name for the output file. A dialogue after starting the program will allow the user to choose a different specific file name for the output data within the above folder. '.mat' will be appended to the specified file name. Currently the following variables are saved in this '.mat' file: 'data_time' (see below), 'data_cmpds' (see below), 'randseed', 'plot_info', 'time_info', and 'cmpd_info'. The last 3 entries are structures that store multiple variables related to their respective titles.

```
plot_info.monitor_cmpds = {'ATP';
          'GTP';
          'MrnaGSHA';
          'GSHA';
          'gammaglutamylcysteine';
          'Cysteine';
          'Glutamate';
          'MrnaGSHA_Ribo';};
```

Stores the time sequence of values for these compounds in data_cmpds{i}(j,k). data_cmpds is a cell vector where each cell contains a 2-dimensional matrix where i specifies the number of the simulation run, row j corresponds to the time in data_time, and column k specifies the compound. Time data (the specific time points at which the compound data is saved) is always saved in data_time{i}(j). Compounds not specified in plot_info.monitor_compounds do not have their time sequence data saved.

```
plot_info.plot_cmpds{1} = {'ATP' 'GTP'};
plot_info.plot_cmpds{2} = {'MrnaGSHA' 'GSHA' 'MrnaGSHA_Ribo'};
plot_info.plot_cmpds{3} = {'gammaglutamylcysteine'};
plot_info.plot_cmpds{4} = {'Glutamate' 'Cysteine'};
```

Specifies which compounds to plot and which figures contain which compound. Each plot_info.plot_cmpds{number} is in the same figure window. One can use plot_info.plot_cmpds = {'all'} if you want each monitor_cmpds to be in own figure window. Each compound to be plotted must also be listed in plot_info.monitor_compounds .

```
plot_info.plot_interval = 200;
```

plot_interval = -1 means do no plots.

plot_interval = 0 means plot at end of each successful timestep (so most likely the time axis won't be at regular intervals)

plot_interval = any other positive number means make a plot when time exceeds the next multiple of the positive value.

plot_interval = any negative integer other than -1 means plot at each timestep that is a multiple of the abs value of this value (i.e. -3 means create the plot at the end of every 3 successful timesteps

Finally, a separate program called 'bns_plot.m' contained in the BNS engine directory is able to load the data stored in the output file (as specified in output_filename above) and recreate the plots that were generated during the simulation run.

Reaction Types
Several reaction types have been defined and the code is designed to be extensible so that more reaction types can be defined. Each reaction requires two files to specify. The model reaction structure is specified in reactions_0001.m, for example. This file specifies the left and right side of a reaction, and the type of reaction. The second file (for example reaction_constants_0001.m) specifies the numerical constants associated with that reaction, typically the reaction rate parameters and the reaction group for that reaction (see next section).

Four reaction types have been defined so far: (1) Nth-order; (2) binding; (3) Transcription and Translation, version 1; and (4) Generalized Michaelis-Menten. It should be noted that strictly speaking, only the binding reaction is valid in all cases when using a Gillespie stochastic algorithm. However the use of single reaction equations to simulate multiple real-world reaction events has been considered (Rao CV and Arkin AP, J Chem Phys **118**:4999-5010, 2003) and is an area of current research.

The following is a summary of the defined reaction types with an example of how the file specifies each reaction. In the descriptions below, a_i is the propensity term for the Gillespie algorithm. It is proportional to the rate at which a reaction proceeds and has units of s^{-1}.

Nth-order
The general form for an n-th order reaction is

$$N_A\, A^q + N_B\, B^r + N_C\, C^s \quad \underset{k_r}{\overset{k_f}{\rightleftarrows}} \quad N_D\, D^t + N_E\, E^u + N_F\, F^v \qquad \text{(Eq. 1)}$$

where A, B, C, D, E, F are compound names; N_A, N_B, N_C, N_D, N_E, and N_F are numbers of each compound consumed in the reaction; q, r, s, t, u, v are exponents reflecting the contribution of each compound to the rate, and k_f and k_r are the forward and reverse rate constants. The units for k_f and k_r will depend on the total order of the corresponding substrates, where the total order n for the forward reaction is q+r+s, and the total order for the reverse reaction is t+u+v. If the total order is 1, then the units for k are s^{-1}. If the total order is 2, then the units for k are liter/s. There can be any number of substrates on the left or right side, and the orders do not have to be integer.

The nth-order reaction is a generic, flexible format that can be used to specify any transformation of one set of compounds to another. It is used when there isn't a more specific reaction type available for a particular reaction.

The above designation creates 2 reactions. The propensity for the forward reaction is given by

$$a_i = k_f * NT_A{}^q * NT_B{}^r * NT_C{}^s \,/\, vol^{(n-1)}, \qquad \text{(Eq. 2)}$$

where n = q+r+s and NT_A is the total number of molecules of compound A in the system (not the total number used for a single execution of this reaction N_A), etc.

The propensity for the reverse reaction using D, E, F follows the same form.

The following is a section of code from a reactions_001.m file which specifies an n-th order reaction

```
    rxDegrade_MrnaGSHA.type = 'nth-order';
    rxDegrade_MrnaGSHA.lhs = {      1 'MrnaGSHA' 1;
                                    1 'RNase' 1;};
    rxDegrade_MrnaGSHA.rhs = {      377 'UMP' 0;
                                    369 'CMP' 0;
                                       381 'AMP' 0;
                                       429 'GMP' 0;
                                       1 'RNase' 0;};
```

The name of the reaction is 'rxDegrade_MrnaGSHA'. This is the part before the period in the Matlab structure above.

Then there are 3 subfields for this structure, '.type', '.lhs', and '.rhs'.

The '.type' subfield is used to specify a string designating the reaction type, which is 'nth-order' in this case.

The '.lhs' subfield is used to specify the left hand side list of reaction components. It is filled in with a cell array (as indicated by the curly braces). Each row of this cell array has 3 values. The first value is the number of that compound that is consumed or produced when the reaction executes, and corresponds to N_A etc. For example, 377 molecules of UMP are produced when the reaction proceeds from left hand side to right hand side. The second value in each row is a string designating the name of the compound. The third value of each row is the order of that compound. Note that these orders are *relative* orders. They are only relative to each other in this file. The exact value of q, r, s, t, u, v to be used in Eqs. (1) and (2) is determined by scaling each of the orders specified in this file so that the total order equals the total order specified in the reaction_constants.m file as discussed below.

The '.rhs' subfield specifies the right hand side list of reaction components in a manner identical to the '.lhs' subfield.

The following is a section of code from a reaction_constants_0001.m file for the above n-th order reaction.

```
rxDegrade_MrnaGSHA.ltor.rate = 1e-19;
rxDegrade_MrnaGSHA.ltor.order = 2;
rxDegrade_MrnaGSHA.rtol.rate = 0;
rxDegrade_MrnaGSHA.rtol.order = 0;
rxDegrade_MrnaGSHA.rxn_grp = 1;
```

The name of the structure (the part before the period above) must match the name of the reaction from the reactions_0001.m file.

The '.ltor' subfield specifies constants for the left-to-right reaction, or the '.lhs' to '.rhs' direction using the nomenclature above. The '.rtol' is for right-to-left.

Each direction has a rate value which corresponds to k_f or k_r in Eqs. (1) and (2). Note that in the example above, '.rtol.rate' is set to zero, meaning the reaction does not proceed from right-to-left, and in fact the code parses it such that this reaction is not even entered into the list of possible reactions.

Each direction has a total order value which corresponds to either $n = q + r + s$ for the left-to-right case, or $n = t + u + v$ for the right-to-left case. As indicated above, this order is the overall order for the reaction, and is used to scale the order values for each compound specified in the reactions_0001.m file so that the total order equals the value specified here.

Finally, the '.rxn_grp' subfield is used to assign this reaction to a reaction group, which will be discussed in the next section.

Binding

The binding reaction type is exactly the same as an nth-order reaction with the restrictions of exactly two compounds on the left side, one compound on the right side, exactly 1 of each compound is consumed or produced, and all the exponents are equal to 1. The reason a separate binding reaction type is created is for execution speed. Binding and unbinding is a very common phenomenon, and the extra code required to execute an nth-order reaction slows the execution of an nth-order reaction compared to this binding reaction type.

$$A + B \underset{k_r}{\overset{k_f}{\rightleftarrows}} C \qquad \text{(Eq. 3)}$$

Creates 2 reactions as long as no rate equals zero. Left-to-right is a 2nd order reaction, and right-to-left is a 1st order reaction. Units for k_f are Liter/s. Units for k_r are s^{-1}.

The propensity function for the forward (binding) case is

$$a_i = k_f * NT_A * NT_B / vol, \qquad \text{(Eq. 4)}$$

and the reverse (unbinding) case is

$$a_i = k_r * NT_C. \qquad \text{(Eq. 5)}$$

An example reactions_0001.m code section is

```
        rxGeneGSHA_T7.type = 'binding';
        rxGeneGSHA_T7.lhs = {'GeneGSHA';
                                        'T7';};
        rxGeneGSHA_T7.rhs = {'GeneGSHA_T7';};
```

The '.type' subfield is 'binding'.

The '.lhs' and '.rhs' subfields are simpler compared to nth-order, and only require the name of the compounds.

An example reaction_constants_0001.m code section for a binding reaction is

```
        rxGeneGSHA_T7.ltor.rate = 1e-18;
        rxGeneGSHA_T7.rtol.rate = 10;
        rxGeneGSHA_T7.rxn_grp = 1;
```

The '.ltor.rate' corresponds to k_f in Eqs. (3) and (4). The '.rtol.rate' is k_r in Eqs. (3) and (5).

Transcription and Translation, version 1

This reaction type is used for transcription and translation reactions. The version 1 indicates that it is a crude, but fast executing, approach for these complex processes.

$$\text{A\_template} + [N_b\, B + N_c\, C + N_D\, D] \xrightarrow{k} N_E\, E + N_F\, F + N_G\, G \qquad \text{(Eq. 6)}$$

Units of k are Liters/s. This reaction type creates a single reaction. The reverse process isn't possible. The first substrate on the left hand side is the template molecule, and only 1 is consumed. The rest of the substrates in the brackets are the building blocks. There can be any number of these building blocks. Typically these are the nucleotides (for transcription) or the amino acids (for translation).

There can be any number of products on the right hand side. Typically the list of products includes a single copy of the primary result of transcription or translation – the messenger RNA or the protein, respectively. It also typically includes some form of the template molecule, although it is not necessarily the identical compound as the A_template from the right hand side due to other changes.

The propensity function used is

$$a_i = k * NT_{A\_template} * (1/\ (1/NT_B + 1/NT_C + 1/NT_D)\ )\ / \text{vol} \qquad \text{(Eq. 7)}$$

where again, NT represents the total number of each compound in the entire system. If, for example, NT_B is less than N_B, then a_i is set to zero. Note that the form for this propensity function is quite arbitrary. It was chosen because it somewhat reflected the fact that a shortage of a single substrate can greatly reduce the overall rate of the reaction.

A reactions_0001.m code section for this reaction follows.

```
rxTransc_GeneGSHA.type = 'transv1';
rxTransc_GeneGSHA.lhs = { 1 'GeneGSHA_T7';
                          377 'UTP';
                          369 'CTP';
                          381 'ATP';
                          429 'GTP';};
rxTransc_GeneGSHA.rhs = { 1 'MrnaGSHA';
                          1 'GeneGSHA';
                          1 'T7';
                          1556 'PPi';};
```

The '.type' is the string 'transv1'.

Each row in the cell array for both '.lhs' and '.rhs' contains two entries. The first entry is the number of molecules consumed or produced by a single execution of this reaction. The second entry is the string for the name of the compound. The first row in the list of '.lhs' is the template molecule and only 1 molecule of that is allowed.

A reaction_constants_0001.m code section follows.

```
rxTransc_GeneGSHA.ltor.rate = 1e-18;
rxTransc_GeneGSHA.rxn_grp = 1;
```

Only two values are needed – the rate which corresponds to k, and the reaction group.

Generalized Michaelis-Menten

This is a generalized Michaelis Menten reaction type. It is intended to mimic the saturation kinetics that occur in a standard Michaelis-Menten enzyme catalyzed equation, but extended to the possibility of more than a single substrate.

$$\text{Enzyme}^{k2f} + N_A\, A^{kmA} + N_B\, B^{kmB} + N_C\, C^{kmC} \rightleftarrows \text{Enzyme}^{k2r} + N_D\, D^{kmD} + NE\, E^{kmE} + NF\, F^{kmF} \qquad \text{(Eq. 8)}$$

The first substrate is the enzyme molecule and only 1 is involved in the reaction. The parameters k2f and k2r correspond to the specific maximum rate of the forward and reverse reaction when the quantities of all other non-enzyme substrates are high. The units of k2f and k2r are liter/s. Note that in an enzyme reaction, the Enzyme usually is both a substrate and a product, although it isn't strictly required that they be the exact same compound.

The rest of the substrates are governed by a km constant which has units of molecules/liter. In other words, at low numbers relative to km * volume, the reaction rate is linear with the number of molecules, and at high numbers relative to km * volume, the reaction rate saturates.

This reaction type creates 2 reactions, unless one of the k2 rates equals zero.

The propensity function for a gen-mm reaction type in the forward direction is:

$$a_i = k2f * (NT_{Enzyme} / vol) * (NT_A / (NT_A + kmA * vol)) * (NT_B / (NT_B + kmB * vol)) * (NT_C / (NT_C + kmC * vol)) \quad \text{(Eq. 9)}$$

where again NT refers to the total number of each compound. The same form is used for the reverse reaction. This propensity function simplifies to the standard Michaelis-Menten rate when the left and right hand sides each contain a single Enzyme and a single other compound (substrate on left-hand-side, product on right-hand-side).

The reactions_0001.m file looks like:

```
rxGSHA.type = 'generalized-mich-ment';
rxGSHA.lhs = {      1 'GSHA';
                    1 'Glutamate';
                    1 'ATP';
                    1 'Cysteine';};
rxGSHA.rhs = {      1 'GSHA';
                    1 'ADP';
                    1 'Pi';
                    1 'gammaglutamylcysteine'};
```

The '.type' string is 'generalized-mich-ment'.

The '.lhs' and '.rhs' subfields follow the same format as the corresponding subfields in the Transcription and Translation, version 1 reaction type. The first row in each cell array is the enzyme molecule, and only 1 molecule of that is allowed.

The reaction_constants_0001.m file looks like:

```
rxGSHA.ltor.rate.k2 = 1e-15;
rxGSHA.ltor.rate.km(1) = 1e19;
rxGSHA.ltor.rate.km(2) = 1e20;
rxGSHA.ltor.rate.km(3) = 1e19;
rxGSHA.rtol.rate.k2 = 0.0;
rxGSHA.rtol.rate.km(1) = 1e19;
rxGSHA.rtol.rate.km(2) = 1e19;
rxGSHA.rtol.rate.km(3) = 1e19;
rxGSHA.rxn_grp = 1;
```

The ‘.rate.k2’ and the ‘.rate.km(n)’ subfields are the k2 and km constants described above.

Reaction Groups

The CPU time required to simulate a complex biological network using the Gillespie algorithm may be very long. A stochastic simulation of biochemical events using the Gillespie algorithm takes place in a defined volume. Following the execution of a single event, the probabilities of all affected events need to be recalculated and then the next event is randomly chosen according to these updated probabilities. One common approach to speed up simulations is to use parallelization. However the Gillespie algorithm does not lend itself well to parallelization, since each event must be executed sequentially.

A summary of the Gillespie stochastic simulation algorithms will aid in understanding the reaction group implementation. The following is Gillespie's Direct Method.

```
Step 0: Initialization. Set time $t = 0$. Calculate the propensity for each reaction, $a_i$,
        based on the number of molecules of the substrates and the reaction rate
        constants.
Step 1: Sum the $a_i$ values from each reaction to create $a_{i,sum}$.
Step 2: Determine the time of the next reaction, $t_{next} = (1/a_{i,sum}) * \log(1/r_1)$, where $r_1$ is
        a randomly generated number between 0 and 1.
Step 3: Determine which reaction $\mu$ is next according to
            $\sum_{i=1}^{\mu-1} a_i < r_2 a_{i,sum} < \sum_{i=1}^{\mu} a_i$,
        where $r_2$ is a randomly generated number between 0 and 1.
Step 4: Adjust the number of molecules according to reaction $\mu$.
Step 5: Set the time $t = t + t_{next}$.
Step 6: Calculate a new propensity $a_i$ for each reaction that was affected by Step 4.
Step 7: Return to Step 1.
```

The following is the hybrid method that is used when stoch_algor = 1 is chosen in general_constants.m. It uses Gillespie's First Reaction Method as a base, but implements reusable randomly generated times as done in Gibson and Bruck's Next Reaction Method.

```
Step 0: Initialization. Set time $t = 0$. Calculate the propensity for each reaction, $a_i$,
        based on the number of molecules of the substrates and the reaction rate
        constants. Determine a putative time for each reaction according to
            $t_{i,putative} = (1/a_i) * \log(1/r_1)$.
Step 1: Let $\mu$ be the reaction with the next $t_{i,putative}$.
Step 2: Adjust the number of molecules according to reaction $\mu$.
Step 3: Set the time $t = t_{\mu,putative}$.
Step 4: Calculate a new propensity $a_i$ for each reaction that was affected by Step 2.
Step 5: Determine a new putative time for each reaction whose $a_i$ changed in Step 4
            $t_{i,putative} = t + (1/a_i) * \log(1/r_1)$.
Step 6: Return to Step 1.
```

An idea to utilize parallelization is to divide the reactions into groups, where each group's reactions are executed on a separate CPU. Each reaction is given a reaction group number in reaction_constants.m. Due to dividing reactions into groups (and hence CPUs), some compounds may be used in multiple reaction groups. The process then is as follows.

Step 0: Initialization. Set time $t = 0$. Choose a value for *timestep*. Calculate the propensity for each reaction, a_i, based on the number of molecules of the substrates N_j and the reaction rate constants. Determine a putative time for each reaction according to

$$t_{i,\ putative} = (1/a_i) * \log(1/r_1).$$

Make a backup copy of a_i and $t_{i,\ putative}$.

Step 1: For each reaction group, set $t_{local} = t$. Also, make a copy of the number of molecules. Call this the local copy of the number of molecules, which is local to each reaction group.

Step 2: For a single reaction group, do the following:

A. Let μ be the reaction with the next $t_{i,\ putative}$ of all the reactions within the reaction group.

B. If $t_{\mu,\ putative} < t + timestep$, then go to step 2C. Otherwise go to Step 2A and repeat all of Step 2 for the next reaction group. After every reaction group has executed Step 2, move to Step 3.

C. Adjust the local copy of the number of molecules according to reaction μ. Keep track of the changes in molecule number that occurred due to events in this reaction group.

D. Set $t_{local} = t_{\mu,\ putative}$.

E. Calculate a new propensity a_i for each reaction that was affected by Step 1C. Use the local copy of the number of molecules in this calculation.

F. Determine a new putative time for each reaction whose a_i changed in Step 1E

$$t_{i,\ putative} = t_{local} + (1/a_i) * \log(1/r_1).$$

G. Return to Step 2A.

Step 3: Total up the changes molecule numbers due to changes in all reaction groups to arrive at a new proposed value for the total number of molecules of each compound, $N_{j,\ proposed} = N_j + N_{j,\ changed}$, where N_j is the number of molecules in Step 1, and $N_{j,\ changed}$ is the total changes in molecule numbers from step 1D for all reaction groups. Note that N_j will be the same for each reaction group that uses compound N. It is a non-local value. N_j is the most recent correct value for the numbers of each molecule, and arises from the conclusion of the most recent successful *timestep* (Step 5).

Step 4: If any $N_{j,\ \text{proposed}}$ is negative or changed "too much" from N_j, then reject all changes that occurred during *timestep*, reduce the value of *timestep*, and go to Step 1. Thus, reject $N_{j,\ \text{proposed}}$ and use N_j. Also let t remain unchanged. Also reset each a_i and $t_{i,\ \text{putative}}$ to the backup copies. If $N_{j,\ \text{proposed}}$ is acceptable, go to Step 5.

Step 5: Set $N_j = N_{j,\ \text{proposed}}$.

Step 6: Set t = t + *timestep*.

Step 7: Increase the value of *timestep* if all $N_{j,\ \text{changed}}$ were "small" relative to N_j in Step 4.

Step 8: Calculate the propensity for each reaction, a_i, based on the number of molecules N_j of the substrates and the reaction rate constants. Determine a putative time for each reaction according to

$$t_{i,\ \text{putative}} = (1/a_i) * \log(1/r_1).$$

It is necessary to do this for all reactions since N_j may have been changed by reaction groups other than the one containing reaction i. Make a backup copy of a_i and $t_{i,\ \text{putative}}$.

Step 9: Return to Step 1.

So in this algorithm, each reaction group operates independently for the duration of *timestep*, using only local information. Once each reaction group has simulated the time from t to t + *timestep*, then the changes in compounds numbers from all reaction groups are totaled and a determination is made as to whether the simulation from t to t + *timestep* was acceptable (more on this determination below). If acceptable, then the process repeats itself. If unacceptable, then *timestep* is reduced and the algorithm tries again to simulate from t to t + *timestep*.

The parallelization concept should be clear, namely the CPU time consuming parts of the simulation – the stochastic event simulation in Step 2 – is divided up among multiple processors. The other steps require the results from all the CPUs to be combined, so are not parallelized, but are not as time-consuming. The larger the value of *timestep*, the fewer times one needs to execute Steps 1 and 3 through 9.

The disadvantage of breaking up the reactions into groups is the loss of exactness in the simulation. Under the assumptions of constant volume, a well-mixed volume, and unchanging reaction rate constants, the Gillespie algorithm produces an exactly correct possible path for the system through time. Exactly correct means that the probability of the system taking each path is correctly simulated according to the assumptions. Thus, whenever more than a single reaction group is used, the algorithm described above is *inexact*.

The use of a variable *timestep* and the choices involved in the determination as to whether a particular simulation from t to t + *timestep* was acceptable allows one to set the degree of inexactness. Limiting *timestep* to smaller values will create a simulation that approaches an exact simulation, but will run no faster (and in fact probably slower) than an exact simulation.

Two conditions were considered to make a simulation from t to t + *timestep* unacceptable. The obvious one is if the total number of molecules of a particular compound becomes negative. The second criteria was if the number of molecules changed too much, which is arbitrarily defined in the current code as increasing by more than a factor of 2, or decreasing by more than a factor of 2 as long as the initial number of molecules was more than 4. The more than 4 molecules part is needed to prevent any change from 1 to 0 or 0 to 1 molecules being flagged unacceptable. A large change in the number of any molecular species increases the inexactness of the simulation if that species is used in multiple reaction groups.

The primary idea on how to divide up the reactions into different groups is that reactions that frequently "communicate" with each other are grouped together. Reactions communicate whenever the substrate or product in one reaction participates in the other. Using this approach to grouping the reactions will minimize the inexactness. Future plans include implementing spatial reaction groups. Here, each reaction group would represent the molecules and reactions that exist in a sub-volume of the entire system.

Complexes

The final code does not incorporate the concepts in this section. Nevertheless, significant effort went into the ideas and implementation of complexes in stochastic simulations. So I feel it is worth briefly documenting these ideas. The use of complexes in stochastic simulations remains an important and unsolved issue in the biological stochastic simulation research community.

Motivation

Complexes of molecular species play a major role in the function of biological cells. The most prominent is the binding of transcription factors to the regulatory elements of a gene. Multi-state proteins and multi-protein complexes are also common. When the individual transcription factors or proteins in a complex also have the capacity to have sub-states (e.g. phosphorylation, non-covalent binding of small molecules), the number of states for the complex grows exponentially.

A single gene with the ability to bind 10 transcription factors has 210, or 1024, different possible states, resulting in potentially 1024 different transcription rates. Also, since each state can experience 10 different binding reactions (one for each transcription factor), there are potentially 10,240 reaction rates to specify. Ultimately, any cell simulation software must define the multiple possible reactions for each state of such a complex. To our knowledge, other efforts at cell simulation (Biospice, SBW, ECell) haven't fully dealt with this issue yet, and require the user to explicitly define these reactions, calling each state of the complex a unique name.

Probably, some redundancy will exist and not every permutation of binding possibilities produces a unique effect for transcription and other binding. However, one assumption made to reduce the scale of the problem when developing cell simulation code, that transcription factors act in a linear or additive manner, is unlikely to be true (Arnone, M.I. and Davidson, E.H. Development, **124**: 1851, 1997).

To use complexes in code for cell simulation, three issues must be resolved:

1. Specification of the state of the complex.
2. Specification of the overall function (such as transcription rate, translation rate, etc.) of the complex for each state.
3. Specification of the binding rates for other elements of the complex for each state.

The use of binding sites as an aid to implementing complexes in cell simulation code is explored below.

Specifying the State of the Complex

The most straightforward approach is to define each state of a complex as a separate species with a unique name. This approach works well if the number of *possible* states for the complex is not too large. Even if the complex involves a large number of discrete elements, if they can only assemble in a fairly restricted sequence, then the number of possible states may be manageably small. An example of a suitable complex would be the phosphorylation state of a protein.

A variation on using unique names for each state is to enumerate the states. For example, a protein with 2 phosphorylation sites will have states numbered 1 through 4.

Regarding binding sites, consider the following hypothetical complex:

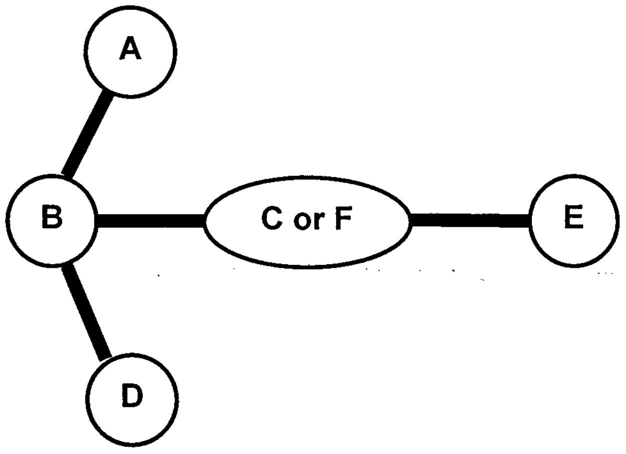


Figure 1
Each letter represents a molecule, and the lines represent possible binding reactions.

If each binding reaction can occur independently, then even counting the number of possible states for this complex is somewhat challenging. (There are 21, not counting states with only a single molecule). The definition of a complex is also an issue. Is **C-E** a permutation of the same complex as **A-B**?

If we use a simplifying assumption that **A**, **D**, and **E** can only exist in unmodified form—i.e. they don't bind to anything other than what is indicated in Fig. 1—then it isn't necessary to name the binding sites on them. The binding sites for this complex would then be: site_on_**B**_for_**A** (call this **B**_st_**A** for brevity), **B**_st_**D**, **B**_st_**CF**, **C**_st_**B**, **F**_st_**B**, **C**_st_**E**, **F**_st_**E**, so a total of 7 binding sites. So complex **A-B** would be specified as follows, with 'emp' signifying an empty site, 'np' signifying a site that isn't even present, and listing the status of the seven sites in the order above: **A**, emp, emp, np, np, np, np. The '**A**' is needed, instead of simply saying 'occupied' for cases like **B**_st_**CF**, where a site can bind multiple molecules. What is the advantage of the 7 element binding site specification as opposed to the name **A-B**? Each approach will need 21 unique identifiers, and the binding site name is longer. One advantage is that one doesn't need to actually name and count all the states. The possible states that can arise will be determined by the possible binding reactions. Another advantage is that the specification itself contains information about which reactions are possible. For example, if **B**_st_**CF** is occupied by **C**, then it is clear (and easy to code) that this instance of the complex can't bind **F**.

Specifying the Function of the Complex

To quantify and code the overall function of the complex, there is no shortcut to specifying a different rate for each permutation that requires one. However, when redundancy exists, that is when multiple permutations produce the same rate, then the use of a binding site specification may be more convenient. For example, if one has an enumeration of all the possible states, then the coding would say something like "*states 1, 3, 4, 7-12, and 17 have rate x*". The binding site specification for the same situation could be "*if **B_st_A** is occupied by **A**, and **B_st_D** is empty, and all the other binding sites can be in any state, then the rate is x*". The binding site specification is in fact a "partial state" specification—only those aspects of the complex that determine the rate of the process are specified. It isn't necessarily shorter, but it does allow the possibility for specifying reactions that are based on biological knowledge.

Specifying the Complex Formation Reactions

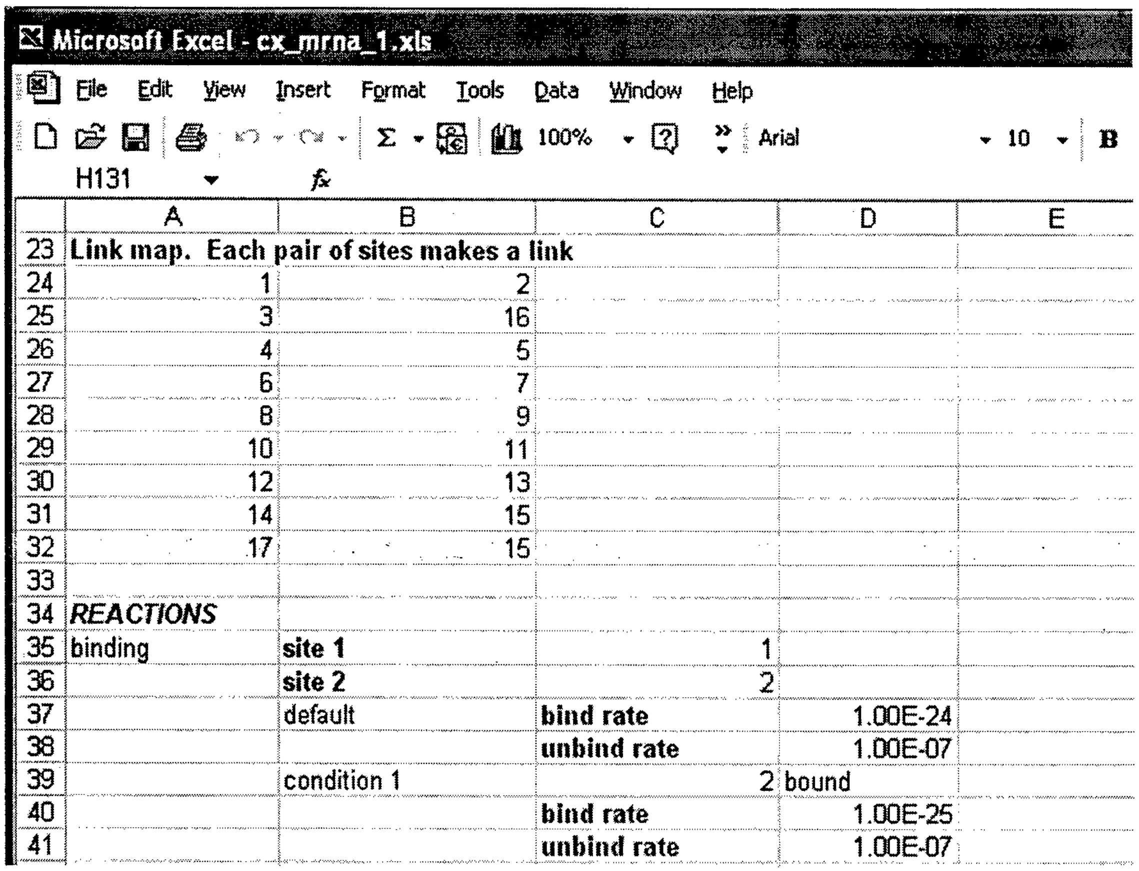
The coding approach for forming the complex is quite similar to those for the (typically) single, overall function of the complex, except that more than one process is likely to be possible for each permutation of the molecule. In other words, for any given state, there will be several possible binding or unbinding reactions. Specifying the partial state of the complex, such as saying a binding site is open, will again assist with constructing the model.

Model Specification vs. Execution

Much of the benefit of using binding sites arises in the initial specification of the model. The concept of binding sites is both intuitive to most people and familiar to biologists, more so than enumerating permutations. The importance of being able to actually describe the model in code should not be underestimated, as this is a very large undertaking. However, the impact of using binding sites on code execution also must be considered. It seems likely that while the partial state specification simplifies some reaction descriptions, it forces the code to check through the existing states of the complex to see if the partial state exists, thereby shifting some of the coding burden to execution time instead of the pre-execution model description. A solution is to have a pre-processing step that takes the binding-site model specification and translates it into an enumerated state specification. Finally, this approach may be very useful if future molecular dynamics simulations are fruitful in estimating binding rates between sites.

User Interface

The specification of biological processes involving complexes for modeling purposes will necessarily be involved. To enable a person to describe the biological processes for the model, the ability to use Microsoft Excel was added (Fig 2). In earlier versions, a graphical interface (Simulink), or actual Matlab code was used. But the graphical approach was not able to handle the layers of complexity required, and it is likely that many end-users will not be proficient in Matlab. The spreadsheet approach makes concepts such as reactions whose rates are conditional upon specific states of a molecular complex easier to define.


Microsoft Excel - cx_mrna_1.xls
File Edit View Insert Format Tools Data Window Help
100%
Arial
10
B
H131
fx

	A	B	C	D	E
23	**Link map. Each pair of sites makes a link**				
24	1	2			
25	3	16			
26	4	5			
27	6	7			
28	8	9			
29	10	11			
30	12	13			
31	14	15			
32	17	15			
33					
34	***REACTIONS***				
35	binding	**site 1**	1		
36		**site 2**	2		
37		default	**bind rate**	1.00E-24	
38			**unbind rate**	1.00E-07	
39		condition 1	2	bound	
40			**bind rate**	1.00E-25	
41			**unbind rate**	1.00E-07	

Figure 2

Simulation and Analysis of Prototype CLEs

CLE simulations

Simulation 1

The first was a demonstration of the possibility that stochastic gene expression can have macroscopic effects. Eight identical cells were each filled with 2 genes. One gene produces a protein that fluoresces red, and the other gene makes a protein that fluoresces green. After letting the simulation run, some of the cells acquire a dominant red color, some are primarily green, some are dark due to low expression of both, and some are colors representing a mixture of red and green (Fig. 3).

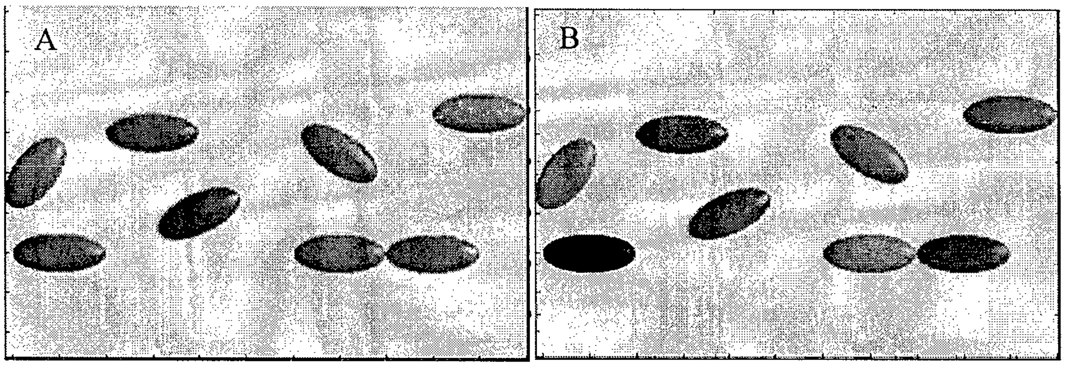


Figure 3: (A) Eight identical cells at time = 0; (B) The same eight cells at time = 3 hrs.

Simulation 2

The second simulation was the initial attempt to design a cell that accomplishes a task. The task was to recognize a specific external molecule, and then to fluoresce blue if that molecule is present. If the external molecule is not present, then the cell fluoresces red. This cell included a membrane bound receptor, signaling molecules that acted as transcription factors, and degradation of the fluorescent proteins (Fig. 4). An animation showing the result in an array of 63 cells was created (Fig. 5). Some of the 63 cells did not contain the blue gene.

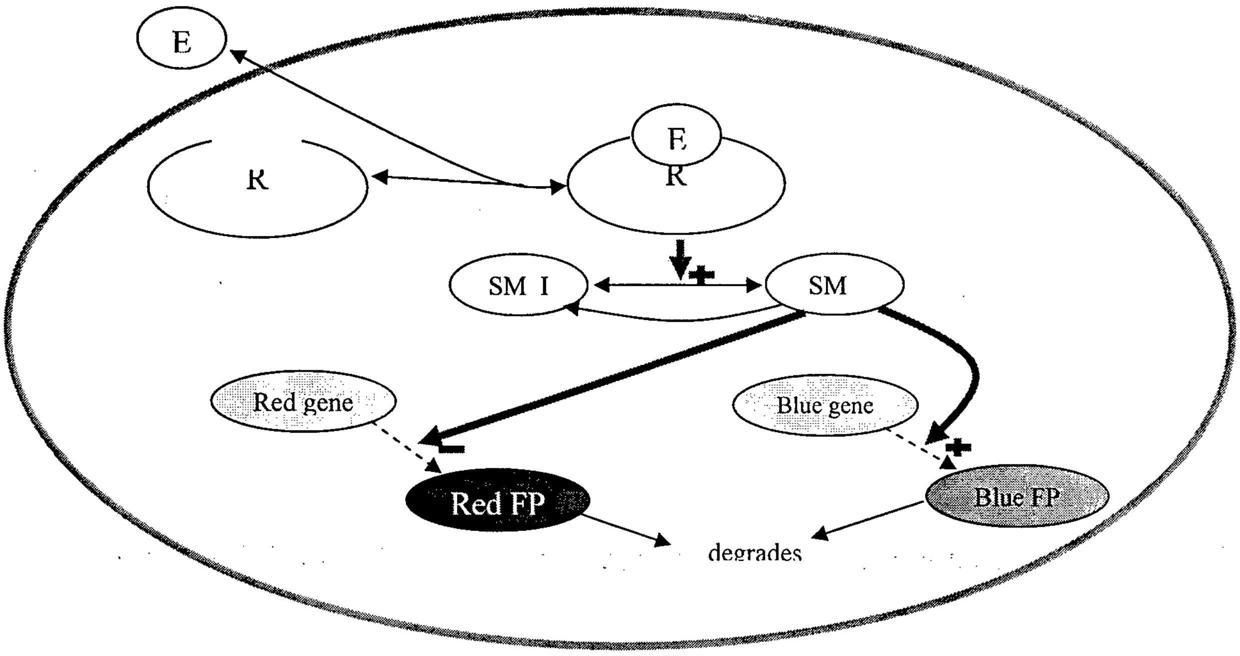


Figure 4: Schematic of a simple detection CLE. E is the external molecule. R is the cytoplasmic receptor. SM is the signaling molecule. SM_I is the inactive form of the signaling molecule. Red_gene and Blue_gene are the genes that produce proteins (no mRNA in this model). Red FP and Blue FP are the red and blue fluorescent proteins. A '+' indicates that the process is accelerated and a '-' indicates the process is slowed.

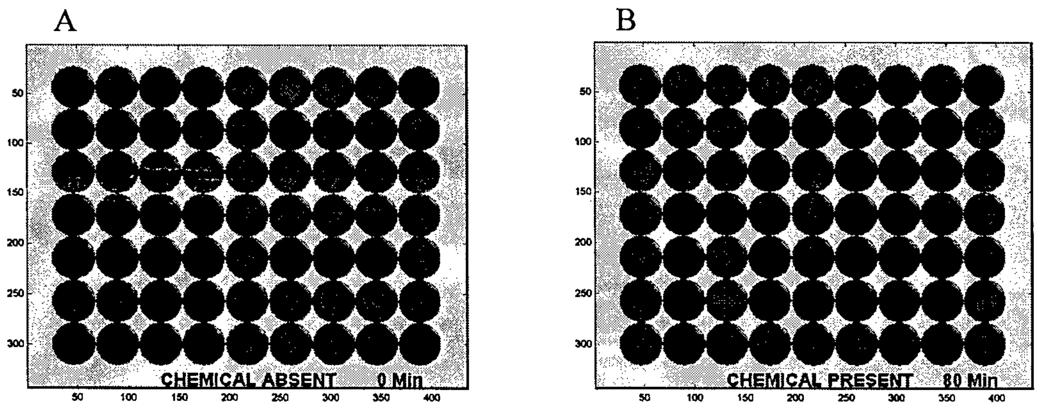


Figure 5: An array of 63 cells containing the biochemical network shown in Fig. 4. Some of the cells did not contain the blue gene. The external chemical ('E') was present from 30 to 80 min. (A) The simulation at time = 0. (B) The simulation at time 80 min.

Simulation 3
This simulation is an extension of the Simulation 2 (Fig. 6). The requirement that the production of proteins requires ATP was added. The generation of ATP from glucose and ADP was added. The glucose is supplied outside the cell and must be transported into the cell to provide the energy. A third gene to produce the membrane bound receptor was added. This model represents an upgrade of the "detection cell-like entity" presented above to one that has energy constraints. Some results of this model are presented in the analysis section.

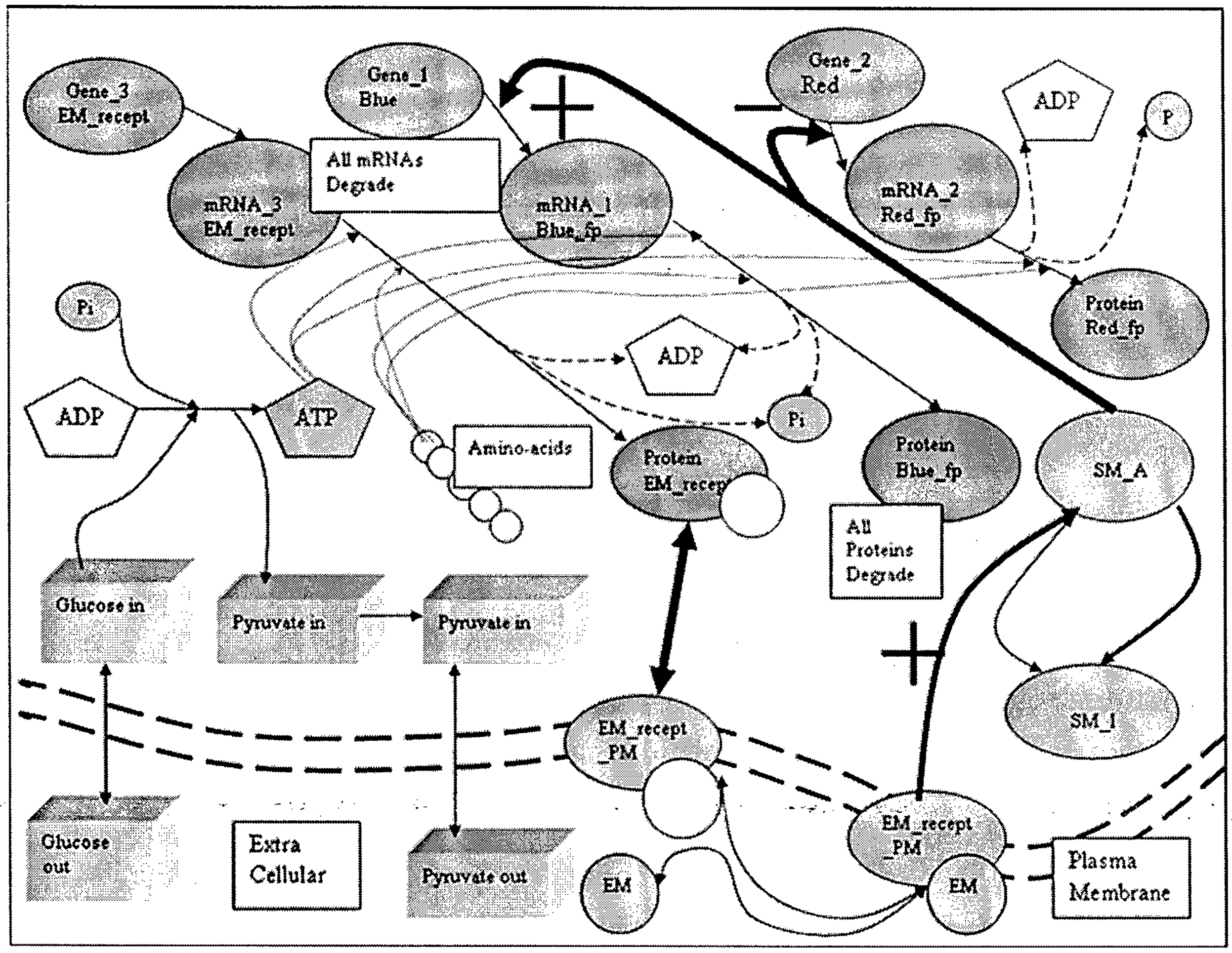


Figure 6: Schematic of a more complete "Detection CLE". Arrows represent reactions, and a + or – symbol represent an enhancement or an inhibition of the reaction rate. EM = External Molecule. Via activated signaling molecule (SM_A), the presence of EM increases expression of blue_fp and decreases red_fp.

Analysis of Stochastic Simulations

Michaelis-Menton study

Both in reviews of a recent proposal, and in discussions with collaborators, the necessity of adding Michaelis-Menton saturable enzyme catalyzed reactions became apparent. There are two ways to implement this type of process. One way is to use the single saturable equation Rate = Vmax*C/(Km+C) where C is the concentration of the substrate. The other way is to model explicitly the two reaction steps involved in a simplified enzyme catalyzed reaction. The rate equation above is derived from an analysis of these two reactions using a couple assumptions. The results are summarized in the graphs below (Fig. 7). The y-axis is the rate of the reaction, and the x-axis is the number of substrate molecules. Parts (A) and (C) represent 2 conditions for the two reaction approach, and parts (B) and (D) represent the same 2 conditions for the single rate

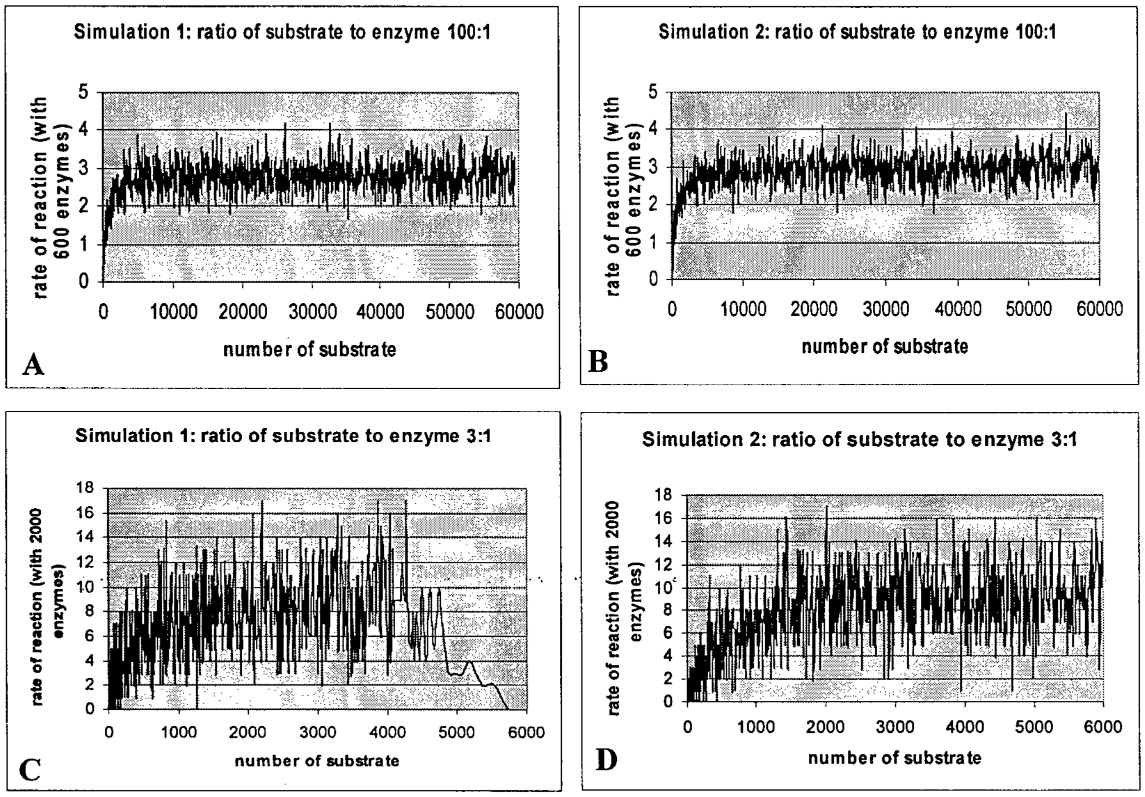


Figure 7: Rate of reaction vs. substrate number. (A) and (C) use the two explicit reaction approach. (B) and (D) use the single familiar Michaelis-Menten rate equation. (A) and (B) have a high substrate to enzyme ratio, whereas (C) and (D) have a lower ratio.

equation approach. One can see that in some cases, (A) vs (B), the approaches produce equivalent results. However in other cases, (C) vs. (D), the results differ and one must use the two reaction approach (C), which is more realistic. This occurs because an assumption used to derive the single rate equation is violated, specifically the assumption that the amount of the substrate-enzyme complex remains at a constant level at all times.

However, the two reaction approach requires 5-10 times as much cpu time, so eventually one would like to find a way to switch between the two approaches depending on whether the assumptions hold.

Baseline Fluctuations

Simulation 3 (Figure 6) as described above was analyzed in a couple of different ways. The first analysis was to run the simulation multiple times under identical conditions to investigate the degree of fluctuations in some key output levels. This simulation entails a period from 0 to 2000 s in which no external molecule is present, then a period from 2000 to 5000 s when external molecule is present, followed by a period from 5000 to 8000 s with no external molecule. Some of these results are expressed in the histograms below. The first one depicts the average level of blue protein during the period when external molecule is present (Figure 8A). The y axis is the number of runs (out of 30 total), and the x-axis is the average number of molecules of blue protein during the time period from 2000 to 5000 s. The second one depicts the average level of red protein during the period from 0 to 2000 s. One can see a large variation spanning almost 400 molecule numbers in the amount of both proteins.

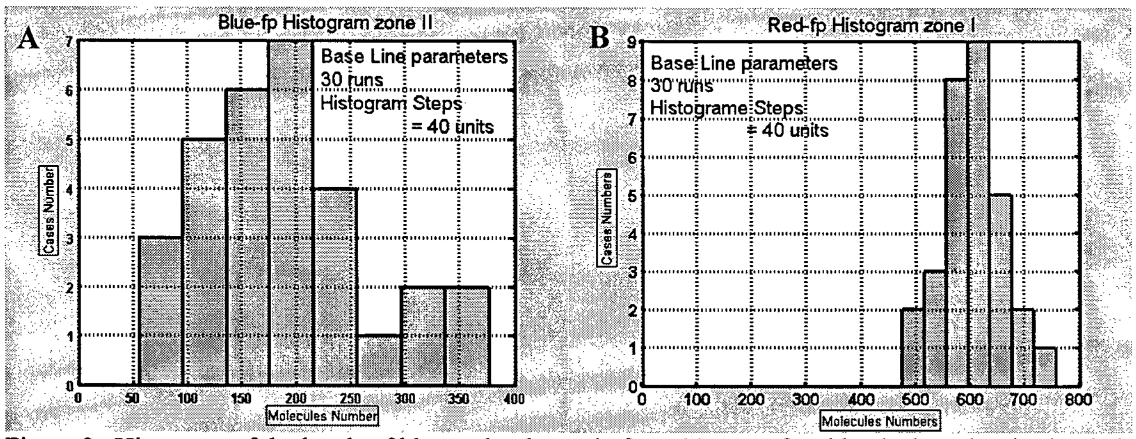


Figure 8. Histogram of the levels of blue and red protein from 30 runs of an identical stochastic simulation. The y-axis is the number of runs. (A) The number of blue protein molecules between 2000 and 5000 s. (B) The number of red protein molecules between 0 and 2000 s.

We also studied the speed of response to the introduction of external molecule. We fit either a polynomial or a spline to the blue signal vs. time curves, and then identified the time required to reach 50% of the average blue signal. The histogram of these response times for the polynomial fit are presented in Fig. 9. Again, the spread in data is quite large, and indicates that serious consideration will need to be given to minimizing fluctuations and variability in the design of such devices if uniform responses are required (which is likely).

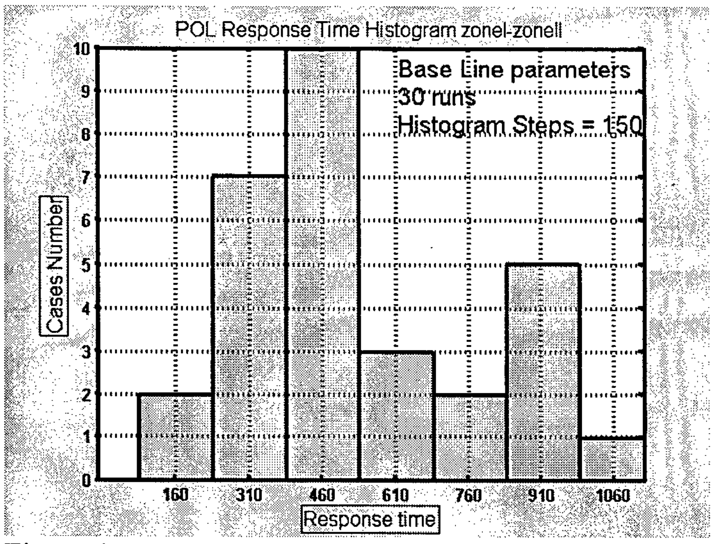


Figure 9

The analysis of stochastic simulations remains in its infancy compared to the analysis of traditional deterministic simulations. This histogram approach is one way to summarize the data.

Parameter Optimization

A second analysis of Simulation 3 was to begin the process of adjusting the parameters of the model so as to optimize the performance of the detection task. We chose 5 parameters to vary, and identified a low and a high value for each parameter. This resulted in 32 permutations, and 5 simulations were run for each permutation. The 5 parameters varied were: (1) number of copies of blue producing gene [1 or 10]; (2) number of copies of red producing gene [1 or 10]; (3) number of signaling molecules [30 or 300]; (4) initial number of EM-receptor molecules [30 or 300]; and (5) degradation rates of mRNA and proteins [low = 0.3 x 10^{-9} s^{-1} for protein and 0.3 x 10^{-6} s^{-1} for mRNA and high = 3 x 10^{-9} s^{-1} for protein and 3 x 10^{-6} s^{-1} for mRNA]. Graphs and summary tables were produced. Figure 10 illustrates the effect of two of the parameters on the amount of blue fluorescent protein. One can see that a high degradation rate (top row) tended to depress the amount of blue signal, as you might expect. Also, one can see that a low number of signaling molecules tended to produce "spikes" in the blue protein signal.

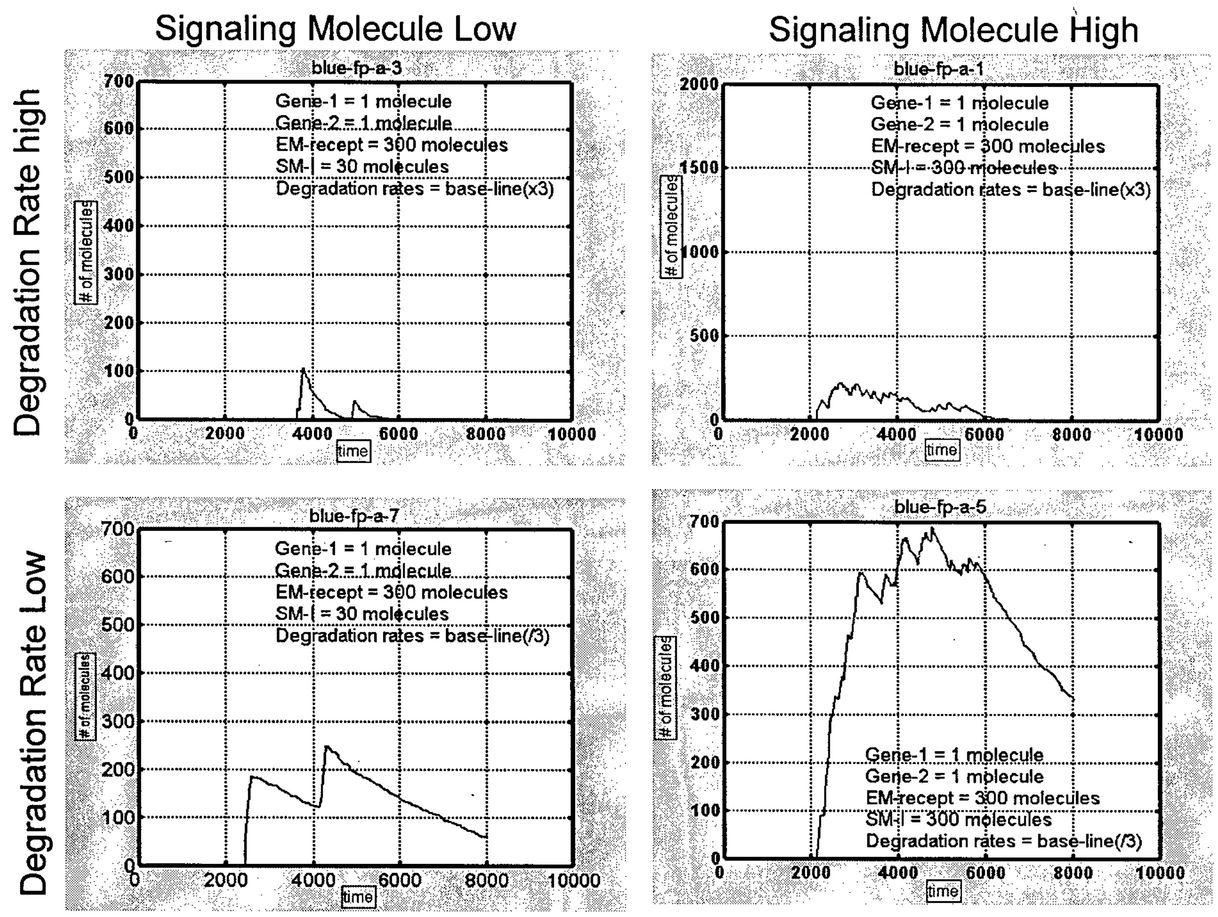


Figure 10

There was one situation which had a nearly optimal response (if we consider an optimal response to be when both the blue_fp/ red_fp ratio is maximum and the response time is minimum). This occurred when number of copies of blue gene was high, number of copies of red gene was low, the number of initial molecules for EM-receptor was low, the number of signaling molecules was high, and the degradation rates were high (Fig. 11). These studies represent our first attempt at optimizing a complex system like a CLE, and we expect to devise more mathematical and efficient optimization strategies in the future.

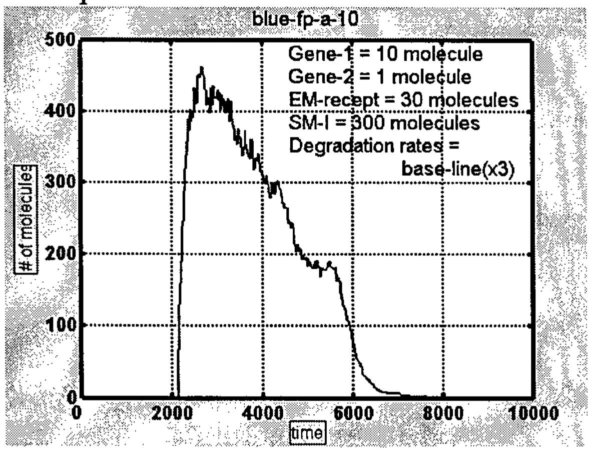


Figure 11

Metabolic Pathway Analysis and Sustained Stress

Bioinformatic Database with Toxicological Emphasis

Since most of the work on this subject is presented in the published manuscript (Karpinets TV, Foy BD, and Frazier JM. Bioinformatics, 20: 507-517, 2004), this summary will be kept brief. A Microsoft Access database on the genes and protein products from the Affymetrix rat toxicology gene chip has been produced. This database divides up the 1031 probes on the gene chip and assigns each one to categories labeled functions, pathways, and location. Function refers to their cellular function, such as detoxification, and each probe is assigned to a single function. Pathway refers to recognized biochemical pathways, and many probes participate in multiple pathways. Location refers to sub-compartment within the cell. Furthermore, the database includes scientific references and annotations to assist with understanding a particular probe's assignment to the above categories. This database has been distributed to several colleagues in the Air Force and at AFIT.

This database was applied to a mathematical clustering of genes from rat hepatocytes exposed to hydrazine. The mathematical clustering was developed to identify those probes that experienced significant changes upon exposure to hydrazine. The hydrazine experiment involved gene chip analysis at multiple time points after dosing. A "Representability Index" was developed to highlight those functions, pathways, or locations that had a significant proportion of probes altered.

Sustained Stress and Cancer Initiation

The above database project led to the development of a schematic model for the major pathways involved in the cell's response to hydrazine. This represents a top-down approach to understanding a cellular reaction network – taking an actual biological system, combining it with large scale biological data from gene chips and protein arrays, and attempting to quantitatively understand the time series of events. Eventually this will be merged with the bottom-up approach of a CLE simulation where each pathway is dynamically simulated.

It was recognized that many of the pathways altered by hydrazine stress are also prominent in tumorigenic transformations. Combining this with the concept of natural selection for rapidly dividing cells in a stressful environment led to the concept of a tumor development model which has been published (Karpinets TV, Foy BD. Carcinogenesis, 26(8): 1323-1334, 2005; and Karpinets TV, Foy BD. Journal of Theoretical Biology, 227: 253-264, 2004). This is a model of numerous interacting biochemical pathways, at both the metabolite and gene levels. The basic statement in the model is that mutations acquired by tumor cells are not caused directly by external DNA damaging agents, but instead are produced by the cell itself as an output of a Mutator Response similar to the bacterial "SOS response" and characterized by the initiation of error-prone cell cycle progression and an elevated rate of mutation. The proposed mechanism is described at the level of involved metabolic pathways and proteins and substantiated by related experimental data available in the literature.

Interactions/Transitions

Presentations

Foy BD, Implementation of Biochemical Complexes in Stochastic Simulations, International Conference on Systems Biology, Heidelberg, Germany, October, 2004.

Kelly-Loughnane N, Thrash ME, Foy BD, Frazier JM. A Gene Expression Model of Glutathione Metabolism in Primary Rat Hepatocytes. International Conference on Systems Biology, St. Louis, November 5-9, 2003 (Abstract and Poster).

Foy BD. Simulating the Interactions of Genes, Proteins, and Metabolites, presented to SPS students, Oct 17, 2003 (Talk).

Foy BD. Mathematical Modeling of Biology, presentation to new Ph.D. students in WSU Biomedical Sciences program. September 23, 2003 (Talk).

Foy, BD. Specification of Molecular Complexes in Cell Simulations. 3rd International Conference on Systems Biology, Stockholm, Sweden, Dec. 2002 (poster presented)

Consultative/Advisory Functions

Served on search committee for opening in the Center for Cell Dynamics and Engineering. This was an international search for a faculty member to assist with the experimental creation of a CLE to be appointed at WSU. Money for this position was from Wright Patterson AFB and State of Ohio.

Regularly attended CLE lab meetings at Wright Patterson AFB and was engaged in additional discussions with Dr. John Frazier about CLE modeling

Transitions

The toxicogenomic database developed in Microsoft Access and published was delivered to Dr. Dennis Quinn of the Air Force Institute of Technology.

Version 1.2 of my cellular simulation tools was delivered to Dr. John Frazier of the Human Effectiveness directorate for use with upcoming initial CLE in a test-tube experiments.

Appendix 1

Main Program Function Hierarchy

All names below are file names without the '.m' extension. The words 'script' or 'function' in parenthesis after the name refers to whether the file is a Matlab script or a Matlab function file. The 'user-edit' in parenthesis indicates that it is a script file that the user is expected to edit to create the model.

bns (script)

- other_constants (script)
    - storing_and_plotting (script, user-edit)
    - general_constants (script, user-edit)
- initial_use_dialogue (script)
- read_cmpds (script)
    - cmpd_defs_0001 (script, user-edit)
    - cmpd_defs_0002, etc. (script, user-edit)
- read_reactions (script)
    - reactions_0001 (script, user-edit)
    - reactions_0002, etc. (script, user-edit)
- read_reaction_constants (script)
    - reaction_constants_0001 (script, user-edit)
    - reaction_constants_0002, etc. (script, user-edit)
- prepare_rxn_groups (script)
    - prepare_binding_rxn (function)
        - build_comps (function)
        - find_comp_id (function)
    - prepare_nth_order_rxn (function)
        - build_comps (function)
        - find_comp_id (function)
    - prepare_transv1_rxn (function)
        - build_comps (function)
        - find_comp_id (function)
    - prepare_genmm_rxn (function)
        - build_comps (function)
        - find_comp_id (function)
- runsim (function)
    - *continued on next page*
- make_plots (function)
- save_data_to_file (function)
- create_cmpd_matrix (function)

- runsim (function)
    - init_rxn_grp (function)
        - create_cmpd_matrix (function)
        - update_rxn_grp (function)
            - update_reaction_probabilities (function)
                - calc_rxn_prob (function)
                    - rxn_prob_binding (function)
                    - rxn_prob_unbinding (function)
                    - rxn_prob_n_order (function)
                    - rxn_prob_transv1 (function)
                    - rxn_prob_genmm (function)
        - save_rxngrp_data (function)
    - store_data (function)
    - decide_if_make_plots (function)
    - make_plots (function)
    - assign_official_data_to_reaction_group (function)
    - execute_reaction_group (function)
        - find_next_reaction (function)
        - execute_reaction (function)
            - execute_binding (function)
            - execute_unbinding (function)
            - execute_normal (function)
        - update_reaction_probabilities (function)
            - calc_rxn_prob (function)
                - rxn_prob_binding (function)
                - rxn_prob_unbinding (function)
                - rxn_prob_n_order (function)
                - rxn_prob_transv1 (function)
                - rxn_prob_genmm (function)
    - tally_changes_in_cmpd_numbers (function)
    - check_last_timestep (function)
    - make_compound_numbers_official (function)
    - update_rxn_grp (function)
        - update_reaction_probabilities (function)
            - calc_rxn_prob (function)
                - rxn_prob_binding (function)
                - rxn_prob_unbinding (function)
                - rxn_prob_n_order (function)
                - rxn_prob_transv1 (function)
                - rxn_prob_genmm (function)
    - save_rxngrp_data (function)